

SCHEDULING DEVICE FOR PERFORMING JOB SCHEDULING OF PARALLEL-COMPUTER SYSTEM

Publication number: JP2002007364

Publication date: 2002-01-11

Inventor: YAMASHITA KOICHIRO

Applicant: FUJITSU LTD

Classification:

- International: G06F15/177; G06F9/50; G06F15/16; G06F9/46; (IPC1-7): G06F15/177; G06F9/46

- European: G06F9/46A4M

Application number: JP20000187317 20000622

Priority number(s): JP20000187317 20000622

Also published as:



US7024671 (B2)

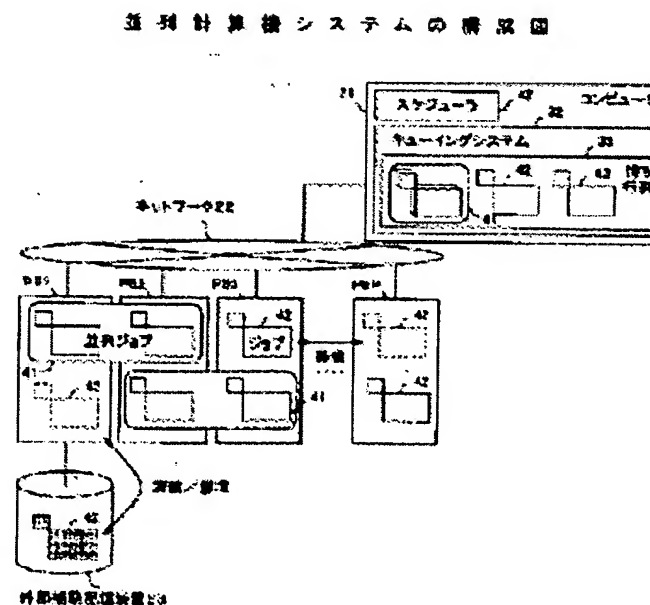
US2002002578 (A)

Report a data error he

Abstract of JP2002007364

PROBLEM TO BE SOLVED: To improve the utilization efficiency of computer resources by reducing a scheduling error accompanying job execution in a parallel-computer system.

SOLUTION: Jobs 41 and 42 inputted from a queuing system 32 to a processor element(PE) are moved to the other PE or frozen and written out to an external auxiliary storage device 23 by dynamic scheduling by a scheduler 31. The scheduler 31 estimates the remaining execution time of the respective jobs and determines the job of a moving/freezing target in comparison with moving/ freezing costs.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-7364

(P2002-7364A)

(43) 公開日 平成14年1月11日 (2002.1.11)

(51) Int.Cl.⁷

識別記号

F I

テーマコード(参考)

G 0 6 F 15/177

6 7 4

G 0 6 F 15/177

6 7 4 A 5 B 0 4 5

6 7 4 B 5 B 0 9 8

6 7 2

6 7 2 B

9/46

3 6 0

9/46

3 6 0 B

3 6 0 C

審査請求 未請求 請求項の数5 O L (全 17 頁)

(21) 出願番号

特願2000-187317(P2000-187317)

(22) 出願日

平成12年6月22日(2000.6.22)

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 山下 浩一郎

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74) 代理人 100074099

弁理士 大菅 義之 (外1名)

Fターム(参考) 5B045 JJ08

5B098 AA10 GA03 GA08 GC10 GD02

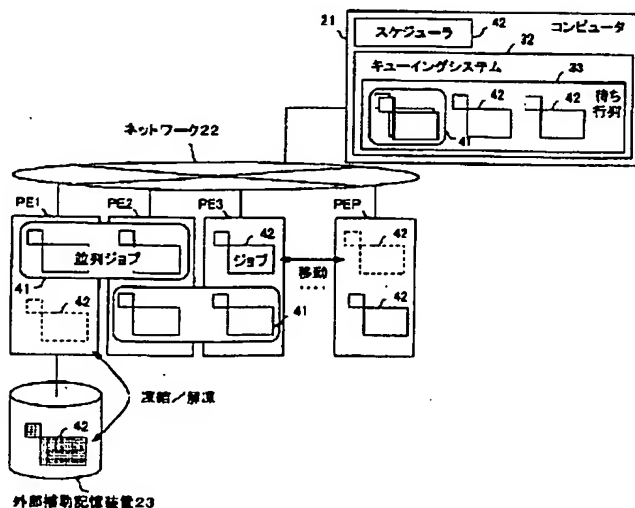
(54) 【発明の名称】 並列計算機システムのジョブスケジューリングを行うスケジューリング装置

(57) 【要約】

【課題】 並列計算機システムにおいて、ジョブ実行に伴うスケジューリング誤差を削減し、計算機資源の利用効率を向上させることが課題である。

【解決手段】 キューイングシステム32からプロセッサエレメント(P E)に投入されたジョブ41、42は、スケジューラ31による動的スケジューリングにより、他のP Eに移動したり、凍結されて外部補助記憶装置23に書き出されたりする。スケジューラ31は、各ジョブの実行残り時間を推定し、移動/凍結コストと比較して、移動/凍結対象のジョブを決定する。

並列計算機システムの構成図



【特許請求の範囲】

【請求項1】 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うスケジューリング装置であって、

プロセッサエレメントで実行中のジョブを他のプロセッサエレメントに移動すべきか否かを判定する判定手段と、

前記ジョブが前記他のプロセッサエレメントに移動するべきであると判定されたとき、該ジョブのマイグレーション処理が行われるように、前記複数のプロセッサエレメントに対する実行中のジョブの割当を行う割当手段とを備えることを特徴とするスケジューリング装置。

【請求項2】 前記複数のプロセッサエレメントの負荷状況を監視する監視手段をさらに備え、前記複数のプロセッサエレメント間で負荷分布がアンバランスになったとき、前記割当手段は、前記実行中のジョブの割当を行うことを特徴とする請求項1記載のスケジューリング装置。

【請求項3】 前記判定手段は、前記実行中のジョブの情報を含むジョブ情報テーブルを生成し、該ジョブ情報テーブルのジョブのうち、移動するジョブを決定して、前記複数のプロセッサエレメント上に再配置されたジョブの情報を含む再配置リストを生成し、前記割当手段は、該再配置リストに基づいて、該実行中のジョブの割当を行うことを特徴とする請求項1記載のスケジューリング装置。

【請求項4】 前記判定手段は、前記ジョブのマイグレーション処理に要するコストを計算し、計算されたコストに基づいて、前記ジョブを前記他のプロセッサエレメントに移動すべきか否かを判定することを特徴とする請求項1記載のスケジューリング装置。

【請求項5】 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うコンピュータのためのプログラムを記録した記録媒体であって、

前記プログラムは、プロセッサエレメントで実行中のジョブを他のプロセッサエレメントに移動すべきか否かを判定し、前記ジョブが前記他のプロセッサエレメントに移動するべきであると判定されたとき、該ジョブのマイグレーション処理が行われるように、前記複数のプロセッサエレメントに対する実行中のジョブの割当を行う処理を前記コンピュータに実行させることを特徴とするコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、並列計算機システムにおいて、動的にジョブスケジューリングを行うスケジューリング装置に関する。

【0002】

【従来の技術】 従来の並列計算機システムにおいては、CPU（中央処理装置）やメモリ等の計算機資源を効率良く利用するために、ジョブのスケジューリングが行われている。ここでは、計算機システムにおいて実行されるプログラムの最小単位をプロセスと定義し、プロセスの複合体として並列実行される単位をジョブと定義することにする。

【0003】 ジョブの実行待ち行列の管理等のスケジューリング方法としては、主に以下のようなものが用いられている。

(1) F I F O (First In First Out) によるジョブの連続投入／実行操作

(2) 優先度による待ち行列内のジョブのソート（スケジューリング）操作

このうち、(2)の方法では、各ジョブに優先度フラグ等を付加しておき、後から待ち行列に投入されたジョブであっても、優先度が高ければ、待ち行列の先頭に割り込ませるような制御を行う。さらに、これらの方法に加えて、ジョブの投入に際して、様々なスケジューリング方法が提案されている。

【0004】

【発明が解決しようとする課題】 しかしながら、上述した従来のスケジューリング方法には、ジョブ投入後の不確定要素に対処できないという問題がある。

【0005】 ジョブが実行される場合、実際の実行時間を実行前に正確に把握することは困難である。これは、OS（オペレーティングシステム）による制御を前提とした場合に、ジョブの実行が、プロセスのスイッチング、あるプロセスと他のプロセスの資源の奪い合い、外部との入出力等の不確定要素を含んでいるためである。

【0006】 この場合、実行時間を正確に管理していないスケジューリング方法によりスケジューリングを行うと、ジョブの実行が進むにつれて、スケジューリングの誤差が大きくなる。その結果、特定のPE（プロセッサエレメント）に負荷が偏る等の事態が発生し、並列計算機システムの運用が非効率的になる。また、OSが複数のプロセスを管理している以上、このような不確定要素によるスケジューリング誤差の発生は不可避である。

【0007】 本発明の課題は、並列計算機システムにおいて、ジョブ実行に伴うスケジューリング誤差を削減し、計算機資源の利用効率を向上させるスケジューリング装置を提供することである。

【0008】

【課題を解決するための手段】 図1は、本発明のスケジューリング装置の原理図である。本発明の第1の局面において、スケジューリング装置は、判定手段11と割当手段12を備え、複数のPEを有する並列計算機システムのジョブスケジューリングを行う。

【0009】 判定手段11は、あるPEで実行中のあるジョブが他のPEにマイグレート（移動）するべきか否

かを判定する。割当手段12は、そのジョブが他のPEにマイグレートするべきであると判定されたとき、そのジョブのマイグレーション処理が行われるように、複数のPEに対する実行中のジョブの割当を行う。

【0010】ジョブのマイグレーション処理とは、あるPEで実行中のジョブを動作させたままで他のPEに移動させる処理を表す。判定手段11は、あるジョブが他のPEにマイグレートするべきか否かを判定し、判定結果を割当手段12に渡す。割当手段12は、そのジョブが他のPEにマイグレートするべきであるという判定結果を受け取ると、そのジョブを他のPEに割り当てるとともに、他の実行中のジョブの再割当を行う。

【0011】このようなスケジューリング装置によれば、ジョブのマイグレーション処理を考慮したジョブスケジューリングが行われるので、ジョブ実行に伴うスケジューリング誤差が発生したときでも、PE上のジョブの配置を適切に変更することができる。したがって、スケジューリング誤差が削減され、計算機資源の利用効率が向上する。

【0012】また、本発明の第2の局面において、スケジューリング装置は、静的スケジューリング手段13と動的スケジューリング手段14を備え、複数のPEを有する並列計算機システムのジョブスケジューリングを行う。

【0013】静的スケジューリング手段13は、静的スケジューリングを行い、動的スケジューリング手段14は、静的スケジューリングと組み合わせられた動的スケジューリングを行う。

【0014】静的スケジューリング手段13は、例えば、静止状態にあるジョブのスケジューリング、または並列計算機システムの構成が固定されている場合のスケジューリングを行う。また、動的スケジューリング手段14は、例えば、動作状態にあるジョブのスケジューリング、または並列計算機システムの構成が変更可能な場合のスケジューリングを行う。

【0015】このようなスケジューリング装置によれば、静的スケジューリングと動的スケジューリングを組み合わせた総合的なジョブスケジューリングが行われる。したがって、ジョブ実行に伴うスケジューリング誤差が発生したときでも、PE上のジョブの配置を動的に変更することで誤差が削減され、計算機資源の利用効率が向上する。

【0016】例えば、図1の判定手段11、割当手段12、静的スケジューリング手段13、および動的スケジューリング手段14は、後述する図2のスケジューラ31に対応する。

【0017】

【発明の実施の形態】以下、図面を参照しながら、本発明の実施の形態を詳細に説明する。本実施形態においては、ジョブスケジューリングに対して、「静的」および

「動的」という概念を取り入れる。そして、ジョブの静止状態あるいはシステムの静止（固定）状態に関連するスケジューリングを「静的スケジューリング」と呼び、ジョブの動作状態あるいはシステムの変動状態に関連するスケジューリングを「動的スケジューリング」と呼ぶことにする。

【0018】ここで、ジョブの静止状態とは、ジョブが実行待ち行列内にある状態や、ディスク等の外部補助記憶装置に格納されている状態（凍結状態）に対応し、ジョブの動作状態とは、並列計算機システム上でジョブが実行されている状態に対応する。また、システムの静止状態とは、システム構成が、運用中に変更されることなく、安定して稼働している状態に対応し、システムの変動状態とは、システム構成が、保守等により運用中に逐次変更される状態に対応する。

【0019】静的スケジューリングにより待ち行列内のジョブを組み替えるだけでなく、動的スケジューリングにより、動作中のジョブを凍結したり、移動したりする操作を考慮したスケジューリングを行うことで、計算機資源をより有効に活用することができる。そこで、本実施形態では、計算機資源を有効に活用するために、以下のような方法を採用する。

(1) 動的スケジューリングにより、実行時間等の不確定要素によるスケジューリング誤差を吸収する。

(2) 動的スケジューリングを効果的に行うために、待ち行列や外部補助記憶装置に対する静的スケジューリングを行う。

(3) 動的スケジューリングを用いて並列計算機システム上のジョブの配置を変えることにより、一部のPE資源を解放し、システムの部分保守を行う。

(4) 動的および静的スケジューリングを組み合わせた総合的なスケジューリングを行う。

(5) 過去のジョブ動作状況のデータを蓄積して、現在のジョブの実行時間を予測する。

【0020】また、スケジューリングによるジョブの実行配置の最適化問題は、一般に、NP困難（non-polynomial hard）な問題であり、最適解を求めるための有効な公式が存在しない。そこで、本実施形態では、経験的（ヒューリスティック）な解法による近似最適解に基づいて、スケジューリングを行う。

【0021】このようなジョブスケジューリングを実現するために、以下のようなハードウェアおよびソフトウェアを備えた並列計算機システムを想定する。・ハードウェア

(1) ジョブを並列に実行可能なPEを複数備えたシステム

(2) 各PEを連結するネットワークシステム

(3) ディスク装置等の外部補助記憶装置とその入出力機構・ソフトウェア

(1) 複数のプロセスを制御（実行）可能なOS

(2) 動作中のジョブを凍結させ、ファイルイメージとして外部補助記憶装置に書き出すようなジョブ凍結機能

(3) 動作中のジョブを別のPEへ移動させるジョブマイグレーション機能

(4) ジョブの投入／実行を待ち行列により管理するキューイングシステム

(5) ジョブの並列度を知るインタフェース機能

(6) ジョブのメモリサイズを知るインタフェース機能

(7) ジョブの実行優先度を指定するインタフェース機能

ここで、ジョブの凍結とは、ジョブの実行を中断して、その時点の動作状態を表すデータ（ジョブに関する一部または全部の情報）をファイルにパッキングし、PEの外部に保存することを意味する。また、ジョブの並列度は、ジョブの実行に必要なPEの数を表し、ジョブのメモリサイズは、ジョブが使用するメモリ量を表す。

【0022】図2は、このような並列計算機システムの構成図である。図2のシステムは、コンピュータ21とP個のPE（PE1、PE2、PE3、...、PEP）を備え、これらの装置はネットワーク22により互いに接続されている。各PEには、必要に応じて、外部補助記憶装置23を接続することができるが、ここでは、PE1に外部補助記憶装置23が接続されている。外部補助記憶装置23としては、例えば、磁気ディスク装置、光ディスク装置、光磁気ディスク（magneto-optical disk）装置、テープ装置等が用いられる。

【0023】コンピュータ21は、スケジューラ31とキューイングシステム32を備える。キューイングシステム32は、実行待ち行列33を含み、実行待ちのジョブ41、42の順序を管理する。スケジューラ31は、上述の方法に従って、動的および静的スケジューリングを行う。各PEは、図3に示すように、CPU51およびメモリ52を備え、ジョブ41、42を実行する。

【0024】キューイングシステム32からPEに投入されたジョブ41、42は、動的スケジューリングにより、他のPEに移動したり、凍結されて外部補助記憶装置23に書き出されたりする。ここで、ジョブ41は、複数のPE上で実行される並列ジョブに対応し、ジョブ42は、1つのPE上で実行されるジョブに対応する。

【0025】また、凍結されたジョブは、その後、外部補助記憶装置23から読み出されて解凍され、動作を継続する。ここで、ジョブの解凍とは、凍結により外部補助記憶装置23に保存されたジョブの情報を、PE上に復元する処理を表す。解凍先のPEは、凍結時の実行PEまたは他のPEである。解凍先のPEが元のPEと異なる場合は、結果的にジョブが移動することになる。

【0026】このように、並列計算機システムにおいて実行中のジョブの凍結や移動を行うことで、計算機資源の不要なアイドル状態等の非効率な状態を改善し、システム本来の性能を引き出すことができる。また、凍結

や移動により特定のPE資源を解放することにより、システムの部分的な保守作業を動的に行うことができる。したがって、並列計算機システムをより効率的に利用することが可能になる。

【0027】図2の並列計算機システムでは、以下の観点から切り分けた各階層においてスケジューリングを行い、ジョブを並べ替える。

(1) 第1階層

システム立ち上げ時の逐次処理（システム立ち上げから待ち行列がジョブで飽和するまで）

(2) 第2階層

待ち行列内におけるジョブの静的スケジューリング

(3) 第3階層

待ち行列からPEへのジョブ投入時の動的スケジューリング（PEと外部補助記憶装置間の入出力）

(4) 第4階層

PE上の実行中のジョブの動的スケジューリング

以下、図4から図16までを参照しながら、第1～第4の各階層におけるスケジューリングについて詳細に説明する。

【0028】図4は、第1階層におけるスケジューリングを示している。この階層では、システムが立ち上がった状態、すなわち、待ち行列33内に後続のジョブが存在せず、計算機資源に空きがある状態を扱う。この場合、スケジューラは、FIFOによりジョブを逐次投入するようなスケジューリングを行う。

【0029】図4では、PE1およびPE2上で並列ジョブ41が実行されているときに、ジョブ42がキューイングシステム内の待ち行列33に投入される。しかし、PE3～PEPの資源が空いているので、ジョブ42は直ちに実行される。このように、システムは、待ち行列33に投入されるジョブを逐次実行可能な状態にある。

【0030】次に、図5は、第2階層におけるスケジューリングを示している。この階層では、第1階層にて逐次処理を続けた後、空き資源が不足して待ち行列33にジョブがたまっている状態を扱う。この場合、PE上のいずれかのジョブが終了し、次のジョブが実行可能になるまでに、スケジューラは、待ち行列33内のジョブを並べ替えて、次に最も有効に投入可能なジョブを、待ち行列33の先頭に持ってくるようにする。

【0031】ここでは、スケジューラは、次のような優先順序で待ち行列33内のジョブをソートする。このソート処理は、例えば、待ち行列33に新たなジョブが投入される度に行われる。

(1) 優先度（緊急度）の高い順にソート

(2) 並列度の小さい順にソート

(3) メモリサイズの小さい順にソート

このようなソート処理によれば、まず、優先度によるジョブの分類・ソーティングが行われ、次に、同じ優先度

のジョブが、並列度／メモリサイズの小さい順にソートされる。言い換えれば、優先度が高く、かつ、小規模なジョブがなるべく早期に実行されるように、ジョブが並べ替えられる。

【0032】この理由は、経験的に、並列度／メモリサイズの小さなジョブは比較的执行時間が短く、また、ジョブの凍結や移動の際にも、操作に必要な時間が短いからである。このように、実行時間が短く機動性の高いジョブを早期に実行することにより、それらのジョブを比較的早期に終了させることができる。

【0033】次に、図6は、第3階層におけるスケジューリングを示している。この階層では、実際に待ち行列33からPEにジョブを投入し、実行させるタイミングでスケジューリングが行われる。

【0034】この場合、キューイングシステム32は、現在実行中のジョブの情報を管理するジョブ情報テーブル61を保持し、スケジューラは、ジョブの凍結や移動に備えて、次の優先順序でジョブ情報テーブル61内のジョブをソートする。このソート処理は、例えば、各階層でスケジューリングが行われる度に行われる。

- (1) 優先度の高い順にソート
- (2) 並列度の小さい順にソート
- (3) メモリサイズの大きい順にソート

図7は、ジョブ情報テーブル61の例を示している。図7のジョブ情報テーブルには、現在実行中のジョブのジョブ名、識別情報(ID)、優先度、並列度、メモリサイズの差＝待ち行列の先頭のジョブのサイズ

－ジョブ情報テーブルから選択されたジョブのサイズの和

(1)

そして、サイズの差が0もしくは負になれば、この処理を終了する。

【0039】P3：ジョブ情報テーブル61の情報から、各ジョブの推定終了時刻と選択されたジョブの凍結処理に必要な時間を計算し、並べ替えオーバーヘッドにより総実行コスト（総実行時間）が増加するか否かをチェックする。そして、総実行コストが増加する場合は、このスケジューリングを中止し、PE上のジョブが自然に終了するのを待つ。ジョブの実行コストの計算方法については、後述することにする。

【0040】P4：総実行コストが増加しない場合、選択されたジョブと待ち行列33の先頭のジョブを入れ替えるのが妥当であると判断し、選択されたジョブを凍結して、外部補助記憶装置23に退避させる。これにより、待ち行列33の先頭のジョブのための資源が確保される。

【0041】P5：待ち行列33の先頭のジョブをPE上に投入し、それをPEに実行させる。図6では、PE1～PE3上のjob_ZZとjob_CCの2つのジョブが選択され、外部補助記憶装置23に退避させられる。そして、空き状態になった資源を利用して、待ち行

イズ、ジョブ投入時刻、アイドル時間等の情報が登録されている。このうち、ジョブ投入時刻は、ジョブの実行開始日時を表し、アイドル時間は、凍結処理等により実行が中断されていた時間を表す。

【0035】例えば、第2階層でのスケジューリングが行われると、スケジューラは、待ち行列33の先頭とジョブ情報テーブル61の先頭のジョブを比較し、前者の優先度が後者のそれより高ければ、実行中のジョブを凍結して外部補助記憶装置23に退避させ、資源の一部を空き状態にする。これにより、ジョブ情報テーブル61のジョブが並べ替えられ、待ち行列33の先頭のジョブが優先的に実行される。

【0036】ここでは、スケジューラは、以下の手順により、退避させるジョブを選択する。この手順においては、ジョブの並列度とメモリサイズを合わせて、ジョブのサイズと呼ぶことにする。

【0037】P1：ジョブ情報テーブル61の最後尾から検索し、待ち行列33の先頭のジョブが必要とするサイズを越えないような実行中ジョブのうち、最も大きなサイズのジョブを選択する。そのようなジョブが存在しなければ、ジョブ情報テーブル61の最後尾のジョブを選択する。

【0038】P2：待ち行列33の先頭のジョブのサイズと、ジョブ情報テーブル61から選択されたジョブのサイズの差を、次式により求め、それを必要なサイズとして用いて、P1およびP2の処理を繰り返す。

列33の先頭のジョブ22が実行される。

【0042】このようなスケジューリングによれば、待ち行列33内の優先度の高いジョブを、より優先度の低い実行中ジョブに優先して実行することができる。また、ジョブ情報テーブル61のジョブを優先度／並列度／メモリサイズによりソートしておくことで、退避に適したジョブを容易に選択することができ、最も効率の良い退避操作を行うことができる。

【0043】次に、第4階層におけるスケジューリングについて説明する。この階層では、スケジューラは、PEの負荷分散とシステムの保守を目的として、実行中のジョブと凍結により退避中のジョブのスケジューリングを行う。これらのジョブは、上述したジョブ情報テーブルに登録されている。ここでは、動作中のジョブを動作させたままシステム内の別のPEに移動させるマイグレーションと、運用中におけるシステム構成の変更とを想定して、動的スケジューリングを行う。

【0044】例えば、CPU資源やメモリ資源を多く消費するような、処理の重いジョブがあるPEに集中した場合、資源に余裕のある他のPEにジョブを移動することで、各PEの負荷を平均化することができ、システム

全体の処理時間を短縮することができる。

【0045】図8は、システム運用中の負荷分布の例を示している。この例では、PE1～PE6の6個のPEが稼働しており、これらのPEがそれぞれ2個のジョブ42を実行している。ジョブ42の矩形内の数字は、そのジョブの負荷量を表し、PEの矩形内の数字は、そのPEの負荷量を表す。PEの負荷量は、そのPE上のジョブの負荷量の総和に相当する。例えば、PE1上の2つのジョブ42の負荷量は、それぞれ“10”および“5”であり、PE1の負荷量は“15”である。

【0046】次に、時間の経過とともに*印のジョブが終了すると、図9に示すような状態になる。この状態では、PE1およびPE2の負荷が重く、PE5およびPE6の負荷が軽い（負荷が“0”である）ため、全体としてPE負荷がアンバランスになっている。

【0047】そこで、マイグレーションを用いた動的スケジューリングにより、一部のPE負荷を分散させることを考える。例えば、図9のPE1およびPE2上の負荷量“5”のジョブ42を、それぞれPE5およびPE6上に移動させるようなスケジューリングを行えば、図10に示すように、PE負荷が平均化される。

【0048】また、システム構成の変更とは、例えば、稼働中のPEを停止したり、新たなPEを追加したりすることに対応する。特定のPEを保守や節電等の理由で停止したい場合、通常は、そのPEに関連するジョブを強制終了させるか、あるいは、これらのジョブが自然に終了するのを待たなければならない。しかし、マイグレーションを用いた動的スケジューリングを行えば、関連するジョブを最適な場所に移動させることができる。したがって、ジョブの実行を継続しながら、PEを停止させることができ、システム構成の変更に対応して、効率的な運用が可能になる。

【0049】例えば、図10においてPE5およびPE6を停止したい場合、PE5およびPE6上のジョブ42を、それぞれPE3およびPE4上に移動させるようなスケジューリングを行えばよい。これにより、図11に示すように、PE5およびPE6を直ちに停止することができ、さらに、残りのPEの負荷が平均化される。

【0050】ところで、ジョブの凍結やマイグレーションの際には、対象ジョブが使用しているすべての資源を解放することを前提に、スケジューラは、ジョブに関連する以下のデータをパッキングする。

(1) ジョブを構成するプロセスのユーザ空間（実行オブジェクトが置かれたメモリ空間、実行オブジェクトが参照するメモリ空間等）

(2) ジョブを構成するプロセスに関連するOS内の制御表（プロセス管理テーブル等）

次に、凍結の場合は、パッキングしたデータをファイルとして外部補助記憶装置に書き出し、マイグレーション

の場合は、パッキングしたデータを移動先のPEに転送して展開する。このように、ジョブに関連するすべてのデータをパッキングして退避／転送することで、PE上でジョブが占有していた資源が完全に解放される。

【0051】UNIX（登録商標）等のシステムの場合、ジョブを構成する各プロセスには、プロセスを識別するためにシステム内でユニークに設定されたIDが付加されている。このため、凍結／解凍、あるいはマイグレーションの操作により、システム内でプロセスIDの矛盾（衝突）が発生する可能性がある。

【0052】このプロセスIDの矛盾を回避するためには、例えば、特開平（「ジョブ実行システム」）または特開平11-338719（「計算機システム」）に記載された技術を用いればよい。

【0053】特開平11-338719では、ジョブを構成するプロセスに対して、ジョブIDと、そのプロセスを実行するPEの仮想プロセッサIDと、そのPE内のローカルプロセスIDとから生成されたプロセスIDが割り付けられる。したがって、ジョブが退避させられた後に、それを構成するプロセスと同じローカルプロセスIDを持つプロセスが生成されても、プロセスIDの衝突は生じない。

【0054】ここで、第3および第4階層のスケジューリングにおいて用いられるジョブの実行コストおよび移動コストの計算方法について説明しておく。ジョブの実行コストは、そのジョブの実行時間に対応し、一般に、コンパイラがジョブのコンパイル時に実行コストをオブジェクトに埋め込まない限り、実行段階でそのジョブの実行コストを正確に算出することはできない。しかし、キューイングシステムは、ジョブを割り当てる際の指標とするために、ジョブの実行メモリサイズを管理しているので、この値を参考にして実行コストを推定することができる。

【0055】ここでは、以下のような相関関係を利用して、実行コストを算出する。

(#1) 一般に、ジョブの実行コストは、使用するメモリサイズに比例する。

(#2) 同一ジョブの実行には、常に同一の実行コストがかかる。

(#3) 1つのPEに複数のジョブが存在する場合、PE内での逐次実行（プロセススイッチング）によるオーバーヘッドは、PE内に存在するジョブの使用メモリ総量に比例する。

【0056】これらの条件をすべて定量化して、すべてのジョブの総実行コストを算出する。まず、(#1)の条件を基準にして、ジョブのメモリサイズの値 $Mem_{job} [byte]$ と実行コスト T の関係を表すと、次のようになる。

$$T = Mem_{job} \times \alpha$$

(2)

ここで、比例係数 σ は、ジョブのメモリサイズと実行コストの関連を定義するパラメータであり、ジョブの実際の実行時間から決められる。

【0057】次に、(#2)の条件を基準にして σ を決定し、各ジョブのメモリサイズと実行コストの関係を得る。ここでは、過去のジョブ実行コスト履歴として、実

$$T = (1/N) \sum t_j$$

ここで、 \sum は j に関する総和を表す。(2)および

(3)式より、このジョブの比例係数 σ は次のように決

$$\sigma = (1/N \times \text{Mem}_{\text{job}}) \sum t_j$$

ただし、初めて実行されるジョブの場合には、 σ として適当な値を設定しておき、それを随時更新しながらシステムの運用を続けることで、 σ の精度を向上させる。

【0058】また、ジョブの形態により、ジョブ自身のサイズは等しくても、入力されるデータファイルのサイ

$$\delta = (1/N) \sum (T - t_j)^2$$

この δ が一定しきい値以上であれば、それらの実行時間 t_j をデータベースの更新に用いず、 δ が一定しきい値未満であれば、実行時間 t_j を用いてデータベースを更新する。

【0059】以上のコスト計算によればジョブの最短実行コストが得られるが、PE上で複数のジョブが実行される場合、実際には、プロセスのスイッチングに伴うメモリの切り替えによるオーバーヘッドが発生する。そこ

$$M_t = \sum \text{Mem}_{\text{job}}$$

ここで、 \sum はPE内のすべてのジョブに関する総和を表す。この M_t を用いて、オーバーヘッド時間 T_o は、次式

$$T_o = M_t \times \lambda$$

ここで、比例係数 λ は、システムの設定により決められる定数である。この場合、各ジョブの実行コストは、

$$C_{\text{exec}} = T + T_o$$

スケジューリングの計算は、ジョブ実行の裏で並列して行われるため、その計算に要するスケジューリングオーバーヘッドは、ジョブ実行コストにより隠蔽される。

【0062】また、ジョブの投入操作や、凍結/解凍またはマイグレーションによる移動操作が行われた場合、(#3)のオーバーヘッドによりジョブの実行時間が変化する。そこで、ジョブの投入時刻と現在時刻から経過時間を算出し、残りの実行コストを補正する。

【0063】ところで、ジョブの移動処理としては、次の2つの処理が考えられる。

(a) 凍結されて外部補助記憶装置に退避したジョブを、凍結前のPEとは別のPE上で解凍する。

(b) マイグレーションにより、パッキングしたデータ

$$C_{\text{freeze}} = C_{\text{defrost}} = \text{Mem}_{\text{job}} / \text{IO} \quad (9)$$

また、ネットワークの通信性能(データ転送性能)を D T [byte/sec] とすると、マイグレーションに

$$C_{\text{migrate}} = \text{Mem}_{\text{job}} / D T$$

例えば、VPPシステムでは、 $\text{IO} = 60$ [Mbyte/sec] であり、 $D T = 32$ [Gbyte/sec]

際のジョブの実行コストのデータベースを作成し、同一ジョブが N 回実行されたとき、それらの実行時間の平均を T とする。 N 回の各々の実行時間を t_j ($j = 1, \dots, N$) とすると、それらの平均は次式により求められる。

$$(3)$$

められる。

$$(4)$$

ズによって実行コストが変化する場合がある。この場合、実行コストのデータベースの更新に用いるデータの選択基準として、次式により実行時間 t_j の分散値 δ を求め、その値に応じてデータの取捨選択を行う。

$$(5)$$

で、(#3)の条件を基準にして、このオーバーヘッドを計算する。

【0060】システムの設定にもよるが、一般に、オーバーヘッド時間 T_o は、ジョブのメモリ使用量に比例して増大する。そこで、ある時点におけるあるPE内のジョブのメモリ使用量を Mem_{job} とすると、総メモリ使用量 M_t は次式により与えられる。

$$(6)$$

により近似することができる。

$$(7)$$

(2)式の T から次式の C_{exec} に置き換えられる。

【0061】

$$(8)$$

をネットワーク経由で転送する。

【0064】上述したように、ジョブのパッキングデータは、メモリ空間とOSの制御表から構成されている。このうち、OSの制御表はジョブによらずに一定であるので、パッキングに要するオーバーヘッドは、ジョブのメモリ空間のみに依存する。

【0065】(a)および(b)の処理における転送媒体は、それぞれ、外部補助記憶装置およびネットワークである。そこで、外部補助記憶装置のデータ入出力性能を IO [byte/sec] とすると、凍結に要するコスト C_{freeze} と解凍に要するコスト C_{defrost} は、次式で与えられる。

要するコストは、次式で与えられる。

$$(10)$$

である。(9)、(10)式のコストを移動コストとして用いて、動的スケジューリングが行われる。

【0066】また、ジョブの投入時刻から既に時間 t が経過している場合、そのジョブの残り時間は C_{exec-t} となる。したがって、 $C_{exec-t} < C_{freeze}$ または $C_{exec-t} < C_{migrate}$ となるような場合は、そのジョブの移動コストが残り時間より大きいので、動的スケジューリングは行われない。

【0067】以上のコスト計算は、第3および第4階層のスケジューリングにおいて行われるものであるが、これを第2階層のスケジューリングに適用することもできる。この場合、例えば、(3) 式の実行コストを用いて待ち行列内のジョブを評価し、実行コストの小さい順にジョブをソートする。

【0068】第4階層のスケジューリングでは、スケジューラは、ジョブの凍結や移動を考慮して、PE上のジョブを再配置する。このとき、再配置に要するオーバヘッドも考慮することで、システム運用の効率化を図る。ここでは、効率化のために、以下のような操作を行う。

(1) 過去のジョブ実行時の実績データをジョブ実行履歴として蓄積しておき、過去に実行された同一ジョブの実行時間の統計をとる。

(2) ジョブの凍結や移動の操作コストを、操作対象のジョブのメモリサイズから計算して、再配置に要するオ

$$Load = (\lambda_1 / P) \sum M_{PEp}$$

ここで、比例係数 λ_1 は、システムの設定により決められる定数であり、 Σ は P 個の PE についての総和を表

$$\delta_{load} = (1/P) \sum (Load - \lambda_1 M_{PEp})^2 \quad (11)$$

システムは、この分散値 δ_{load} を、あらかじめ決められたしきい値 δ_0 と比較し、 δ_{load} が δ_0 より大きい場合に、負荷状況がアンバランスであると判定する。ここでは、メモリ使用量に関する負荷を監視しているが、代わりに、CPUやデータ入出力に関する負荷を監視してもよい。

【0071】図12は、上述のようなスケジューリングの契機を示すフローチャートである。並列計算機システムは、通常の運用動作を行い(ステップS1)、いずれかのジョブが終了したか否かをチェックする(ステップS2)。いずれのジョブも終了していなければ、次に、現在の時刻が定期的なスケジューリングポイントに対応するか否かをチェックする(ステップS3)。

【0072】現在の時刻がスケジューリングポイントに対応すれば、次に、各ジョブのメモリサイズ71を用いて、各PEの負荷の分散値を計算し(ステップS4)、負荷状況がアンバランスか否かを判定する(ステップS5)。そして、負荷状況がアンバランスであれば、スケジューリングを実施し(ステップS6)、通常の運用動作に戻る。

【0073】ステップS2においていずれかのジョブが終了した場合は、そのままステップS4以降の処理を行う。また、ステップS3において現在の時刻がスケジューリングポイントに対応しない場合、および、ステップ

ーバヘッド時間を見積もる。

(3) 実行時間の統計情報とオーバヘッド時間を比較し、ジョブの再配置が完了する前に終了すると予測されるジョブについては、再配置を行わない。

(4) 再配置を行わないジョブが占有している計算機資源については、スケジューリングの際にマスクしておく。

【0069】このスケジューリングは、例えば、以下のような場合を契機として行われる。

(1) システム保守やPEの故障により、PE構成が変化する場合

(2) PEの負荷状況を監視し、PE間で負荷状況がアンバランスな場合

(3) 一定のインターバルタイムを設定し、定期的にスケジューリングを行う場合

(2) の負荷状況の監視は、(3) のインターバルタイムに基づいて定期的に行われる他、ジョブが終了する毎にも行われる。また、負荷状況は、以下のような方法で判定される。

【0070】 p 番目の PE の総メモリ使用量を M_{PEp} とすると、 P 個の PE からなるシステムの平均負荷 $Load$ は、次式で与えられる。

$$(11)$$

す。この $Load$ を用いて、各 PE の負荷の分散値 δ_{load} は、次式により求められる。

$$(12)$$

S5において負荷のバランスが取れている場合は、スケジューリングを行わずに通常の運用動作に戻る。

【0074】図13は、図12のステップS6で行われるスケジューリング処理のフローチャートである。スケジューラは、まず、ジョブ情報テーブル61に登録されているジョブを、以下の優先順序でソートする(ステップS11)。

(1) 優先度(緊急度)の高い順にソート

(2) メモリサイズの大きい順にソート

次に、ジョブ情報テーブル61から、PE上に再配置されたジョブを含む再配置リスト81を生成する(ステップS12)。次に、各ジョブのメモリサイズ71、各ジョブの実行経過時間72、各ジョブの実行コスト履歴73、外部補助記憶装置のデータ入出力性能74、およびネットワークのデータ転送性能75に基づいて、上述した計算方法により、各ジョブの残り時間を推定し、移動コストを求める(ステップS13)。そして、残り時間と移動コストを比較して、各ジョブが移動可能か否かを判定する。

【0075】こうして、再配置リスト81に登録されたジョブは、移動しないジョブ82、通常ジョブ83、移動するジョブ84、および凍結対象ジョブ85に分類される。このうち、通常ジョブ83は、移動しても、なくてもよいジョブに対応し、凍結対象ジョブ85は、ジ

ジョブの凍結処理86により外部補助記憶装置に退避せられる。

【0076】次に、再配置リスト81のジョブを分類結果に基づいてソートし（ステップS14）、再配置リスト81を最適化する（ステップS15）。そして、最適化された再配置リスト81に基づいてジョブの再割当を行い（ステップS16）、処理を終了する。

【0077】図14は、図13のステップS12で行われる再配置リスト生成処理のフローチャートである。スケジューラは、まず、システム上のPEから保守等のために停止するPE（または、ジョブを割り当てたくないPE）を除外して、残りのPEを含む運用PEリストを生成する（ステップS21）。

【0078】次に、ソート後のジョブ情報テーブルのジョブのうち、除外されたPEに割り付けられているジョブに、「移動するジョブ」のフラグを立てる（ステップS22）。このフラグは、強制的な移動対象のジョブであることを表し、このフラグが立っているジョブは、マイグレーション処理の対象となる。また、PEの停止により強制終了させられたジョブは、待ち行列に追加される。

【0079】次に、ソート後のジョブ情報テーブルの先頭のジョブから順に、運用PEリストのPEに対して、計算機資源の許す限り、仮定的に割り当てていき、PEに割り当てられたジョブを再配置リストに登録する（ステップS23）。そして、PEに割り当てられなかったジョブを凍結対象ジョブに決定して（ステップS24）、処理を終了する。

【0080】図13のステップS13では、スケジューラは、各ジョブの実行コスト C_{exec} と経過時間 t から、終了するまでに要する残り時間 $C_{exec} - t$ を求め、凍結対象ジョブの凍結コスト C_{freeze} を算出する。そして、 C_{freeze} が $C_{exec} - t$ より大きい場合は、そのジョブを凍結対象ジョブから除外し、このジョブの終了に合わせて、再スケジューリングを行う時刻を修正する。

【0081】次に、システム上で実行されるジョブのマイグレーションコスト $C_{migrate}$ を算出し、 $C_{migrate}$ が $C_{exec} - t$ より大きいジョブに、「移動しないジョブ」のフラグを立てる。これにより、移動が完了する前に終了すると予測されるジョブが再配置対象から除外され、そのジョブが使用している資源がマスクされる。

【0082】また、図13のステップS14では、スケジューラは、再配置リストのジョブを、以下の優先順序でソートする。

(1) 「移動しないジョブ」のフラグが立っているグループ、フラグなしのグループ（通常ジョブ）、および「移動するジョブ」のフラグが立っているグループが、それぞれ、リストの前部、中間部、および後部にくるようにソートする。

(2) 各グループ内のジョブをメモリサイズの大きい順

にソートする。

【0083】図15は、図13のステップS15で行われる最適化処理のフローチャートである。スケジューラは、まず、ソート後の再配置リストにおいて、通常ジョブのグループの先頭のジョブの順位を変数 n に代入し（ステップS31）、1番目～ $n-1$ 番目のジョブを移動しないジョブとみなして、元のPEに割り当てる（ステップS32）。

【0084】次に、 n 番目以降のジョブを再配置対象のジョブとみなして、空きPEに割り当て、新たな再配置リストを生成する（ステップS33）。このとき、移動しないジョブにより既に資源を占有されているPEに配慮して、負荷（メモリ使用量）は均等になるように、ジョブを配置する。具体的には、資源の余裕が大きいPE群に対して、メモリサイズの大きなジョブから順番に割り当てる。ここでは、ジョブがメモリサイズの大きい順にソートされているので、ソート順にジョブをPEに割り当てればよい。

【0085】次に、再配置により移動するジョブのマイグレーションコスト $C_{migrate}$ を算出し、それらの最大値 C_{max} を求める（ステップS34）。次に、 $n = n + 1$ とおき（ステップS35）、 n 番目のジョブに「移動するジョブ」のフラグが立っているか否かをチェックする（ステップS36）。そして、 n 番目のジョブが移動するジョブでなければ、ステップS32以降の処理を繰り返す。こうして、複数の新たな再配置リストが生成される。

【0086】ステップS36において、「移動するジョブ」のフラグが現れれば、次に、得られた再配置リストのうち、 C_{max} の値が最も小さいものを、最適な再配置リストとして選択し（ステップS37）、処理を終了する。

【0087】図16は、図15のステップS33で行われる割当処理の例を示すフローチャートである。ここでは、運用PEリストに登録されたPEの数を P としている。スケジューラは、まず、PEの番号を表す変数 p に1を代入し（ステップS41）、 n 番目のジョブの並列度を変数 a に代入する（ステップS42）。

【0088】次に、 $p + a - 1$ と P を比較する（ステップS43）。 $p + a - 1$ が P 以下であれば、 n 番目のジョブに p 番目～ $p + a - 1$ 番目のPEを割り当て（ステップS44）、 $p = p + a$ とおく（ステップS45）。

【0089】次に、 $n = n + 1$ において（ステップS46）、 n 番目のジョブが存在するか否かをチェックする（ステップS47）。 n 番目のジョブが存在すれば、そのジョブについて、ステップS42以降の処理を繰り返す。

【0090】また、ステップS43において $p + a - 1$ が P を越えれば、 n 番目のジョブに、1番目～ $p + a - 1 - P$ 番目のPEと p 番目～ P 番目のPEを割り当てる。

(ステップS48)。そして、 $p = p + a - P$ において(ステップS49)、ステップS46以降の処理を行う。こうして、ステップS47においてn番目のジョブがなくなれば、処理を終了する。

【0091】以上説明したように、本実施形態のスケジューリングには、以下のような特徴がある。

(1) 静的および動的な側面から総合的なスケジューリングを行う。

(2) 過去の実行情報を蓄積することにより、確定不可能なジョブの実行時間を予測し、スケジューリングにフィードバックする。

(3) ジョブの凍結による外部補助記憶装置への退避と、PE間でのジョブの移動等を組み合わせて、より高度なスケジューリングを行う。

(4) PEの停止のような、システム全体のオペレーションを考慮したスケジューリングを行う。

【0092】このようなスケジューリングを行うことで、計算機資源を有効に利用することができる。図2のコンピュータ21は、例えば、図17に示すような情報処理装置に対応する。図17の情報処理装置は、CPU(中央処理装置)91、メモリ92、入力装置93、出力装置94、外部記憶装置95、媒体駆動装置96、およびネットワーク接続装置97を備え、それらはバス98により互いに接続されている。

【0093】メモリ92は、例えば、ROM(read only memory)、RAM(random access memory)等を含み、処理に用いられるプログラムとデータを格納する。CPU91は、メモリ92を利用してプログラムを実行することにより、必要な処理を行う。例えば、図2のスケジューラ31およびキューイングシステム32は、プログラムとしてメモリ92に格納される。

【0094】入力装置93は、例えば、キーボード、ポインティングデバイス、タッチパネル等であり、ユーザからの指示や情報の入力に用いられる。出力装置94は、例えば、ディスプレイ、プリンタ、スピーカ等であり、ユーザへの問い合わせや処理結果の出力に用いられる。

【0095】外部記憶装置95は、例えば、磁気ディスク装置、光ディスク装置、光磁気ディスク(magneto-optical disk)装置、テープ装置等である。情報処理装置は、この外部記憶装置95に、上述のプログラムとデータを保存しておき、必要に応じて、それらをメモリ92にロードして使用する。

【0096】媒体駆動装置96は、可搬記録媒体99を駆動し、その記録内容にアクセスする。可搬記録媒体99としては、メモ리카ード、フロッピー(登録商標)ディスク、CD-ROM(compact disk read only memory)、光ディスク、光磁気ディスク等、任意のコンピュータ読み取り可能な記録媒体が用いられる。ユーザは、この可搬記録媒体99に上述のプログラムとデータを格

納しておき、必要に応じて、それらをメモリ92にロードして使用する。

【0097】ネットワーク接続装置97は、ネットワーク22に接続され、PEとの通信に伴うデータ変換を行う。また、ネットワーク接続装置97は、他の任意の通信ネットワークに接続することもできる。この場合、情報処理装置は、上述のプログラムとデータをネットワーク接続装置97を介して他の装置から受け取り、必要に応じて、それらをメモリ92にロードして使用する。

【0098】図18は、図17の情報処理装置にプログラムとデータを供給することのできるコンピュータ読み取り可能な記録媒体を示している。可搬記録媒体99や外部のデータベース100に保存されたプログラムとデータは、メモリ92にロードされる。そして、CPU91は、そのデータを用いてそのプログラムを実行し、必要な処理を行う。

(付記1) 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うスケジューリング装置であって、プロセッサエレメントで実行中のあるジョブを他のプロセッサエレメントに移動すべきか否かを判定する判定手段と、前記ジョブが前記他のプロセッサエレメントに移動するべきであると判定されたとき、該ジョブのマイグレーション処理が行われるように、前記複数のプロセッサエレメントに対する実行中のジョブの割当を行う割当手段とを備えることを特徴とするスケジューリング装置。

(付記2) 前記複数のプロセッサエレメントの負荷状況を監視する監視手段をさらに備え、前記複数のプロセッサエレメント間で負荷分布がアンバランスになったとき、前記割当手段は、前記実行中のジョブの割当を行うことを特徴とする付記1記載のスケジューリング装置。

(付記3) 前記判定手段は、前記実行中のジョブの情報を含むジョブ情報テーブルを生成し、該ジョブ情報テーブルのジョブのうち、移動するジョブを決定して、前記複数のプロセッサエレメント上に再配置されたジョブの情報を再配置リストを生成し、前記割当手段は、該再配置リストに基づいて、該実行中のジョブの割当を行うことを特徴とする付記1記載のスケジューリング装置。

(付記4) 前記判定手段は、前記ジョブのマイグレーション処理に要するコストを計算し、計算されたコストに基づいて、前記ジョブを前記他のプロセッサエレメントにマイグレートすべきか否かを判定することを特徴とする付記1記載のスケジューリング装置。

(付記5) 前記判定手段は、前記ジョブの実行履歴情報に基づいて、該ジョブの実行コストを推定し、推定された実行コストを用いて、前記ジョブを前記他のプロセッサエレメントにマイグレートすべきか否かを判定することを特徴とする付記1記載のスケジューリング装置。

(付記6) 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うスケジューリング装置であって、静止状態にあるジョブのスケジューリングを行う静的スケジューリング手段と、動作状態にあるジョブのスケジューリングを行う動的スケジューリング手段とを備えることを特徴とするスケジューリング装置。

(付記7) 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うスケジューリング装置であって、前記並列計算機システムの構成が固定されている場合のスケジューリングを行う静的スケジューリング手段と、前記並列計算機システムの構成が変更可能な場合のスケジューリングを行う動的スケジューリング手段とを備えることを特徴とするスケジューリング装置。

(付記8) 複数のプロセッサエレメントを有する並列計算機システムであって、プロセッサエレメントで実行中のジョブを他のプロセッサエレメントに移動するべきか否かを判定する判定手段と、前記ジョブが前記他のプロセッサエレメントに移動するべきであると判定されたとき、該ジョブのマイグレーション処理が行われるように、実行中のジョブのスケジューリングを行うスケジューリング手段とを備えることを特徴とする並列計算機システム。

(付記9) 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うコンピュータのためのプログラムを記録した記録媒体であって、前記プログラムは、プロセッサエレメントで実行中のジョブを他のプロセッサエレメントに移動するべきか否かを判定し、前記ジョブが前記他のプロセッサエレメントに移動するべきであると判定されたとき、該ジョブのマイグレーション処理が行われるように、前記複数のプロセッサエレメントに対する実行中のジョブの割当を行う処理を前記コンピュータに実行させることを特徴とするコンピュータ読み取り可能な記録媒体。

(付記10) 複数のプロセッサエレメントを有する並列計算機システムのジョブスケジューリングを行うコンピュータのためのプログラムを記録した記録媒体であって、前記プログラムは、静的スケジューリングと動的スケジューリングを組み合わせたスケジューリング処理を、前記コンピュータに実行させることを特徴とするコンピュータ読み取り可能な記録媒体。

【0099】

【実施例】次に、図19から図21までを参照しながら、上述した並列計算機システムの動作シミュレーションの結果について説明する。シミュレーション対象のモデルの詳細は、以下の通りである。

(1) PEの数：5，10，50台

(2) ハードウェア：

・外部補助記憶装置のデータ入出力性能 60 [M b y

t e / s e c]

・通信装置のデータ転送性能 32 [G b y t e / s e c]

・各PEのメモリ容量 10 [G b y t e]

(3) ソフトウェア：1PE内4多重並列実行可能OS、キューイングシステム

(4) 投入ジョブ：ポワソン分布に基づくランダムな特性を持った、以下のようなジョブ群

・ジョブの種類 50種類

・並列度 1～10

・メモリサイズ 1～10G

・同一ジョブでの実行コストのばらつき 任意

・ジョブの到着時刻 任意

・投入ジョブ数 1000

・PE構成の変更 任意の時刻に10%のPEを停止し、任意の時刻に元の構成に戻す。

【0100】以上のような同一のジョブ群に対して、以下の3種類のスケジューリングを適用するシミュレーションを行った。

(1) F I F Oによる逐次実行

(2) システムの空き資源を考慮した、待ち行列内のスケジューリング

(3) 上述した実施形態の階層的スケジューリング（負荷の分散値のしきい値 δ 0として3種類の値を使用）

なお、(2)式の σ の初期値は1 [s e c / b y t e]

とし、(5)式の分散値 δ のしきい値は1250 [s e c²]

とし、(7)式の比例係数 λ は0.5 [s e c / b y t e]

とした。ここでは、シミュレーションにおける時間の単位としてs e c（秒）を用いているが、これをクロック数等の他の単位に置き換えても、同様の結果が得られる。

【0101】また、第4階層のスケジューリングを行う契機を規定する(12)式の分散値 δ_{load} のしきい値 δ 0をパラメタとして、 δ 0=25，50，75の3種類の値を用いた。この値が小さいほど、スケジューリングが頻繁に行われることを意味する。

【0102】また、(1)および(2)のスケジューリング方法においてPEの停止を伴う場合、動作中のジョブの実行時間を監視し、ジョブの再投入を行う場合のコストと、ジョブの終了待ちを行う場合のコストを比較し、より有効な操作を選択している。

【0103】以上のようなシミュレーションにより、図19、20、および21のような結果が得られた。図19、20、および21のシミュレーション結果は、それぞれ5、10、および50台のPEからなる並列計算機システムのシミュレーションを表している。

【0104】ここで、総実行時間は、上述のジョブ群を実行するのに要した時間を表し、性能比は、(1)のスケジューリング方法の総実行時間を1としたときの他のスケジューリング方法の総実行時間の割合を表し、メモ

リ使用率は、ジョブ群が使用したメモリの割合を表す。これらの測定結果および実行状態から、以下のような傾向が読み取れた。

(1) システムが大規模になる(P Eの数が増加する)につれて、スケジューリング方法の違いによる性能差が小さくなる。

(2) 第4階層のスケジューリングにおいて、P Eの数が少ない場合は、しきい値 $\delta 0$ を小さな値に設定して、スケジューリング頻度を高くすると効率がよい。また、P Eの数が多い場合は、しきい値 $\delta 0$ を大きな値に設定して、スケジューリング頻度を低くすると効率がよい。

(3) 運用中にP Eが停止した場合、(1)および(2)のスケジューリング方法では、ジョブの再投入または終了待ちによる処理の遅延が目立った。言い換えれば、ジョブの移動による継続運用が有効であることが分かった。

【0105】以上のような傾向から、(3)の階層的スケジューリングが有効であることが分かる。

【0106】

【発明の効果】本発明によれば、並列計算機システムにおいて、動作中のジョブの凍結や移動を考慮した動的スケジューリングを行うことで、ジョブ実行に伴うスケジューリング誤差を削減し、計算機資源の利用効率を向上させることができる。

【図面の簡単な説明】

【図1】本発明のスケジューリング装置の原理図である。

【図2】並列計算機システムの構成図である。

【図3】P Eの構成図である。

【図4】第1のスケジューリングを示す図である。

【図5】第2のスケジューリングを示す図である。

【図6】第3のスケジューリングを示す図である。

【図7】ジョブ情報テーブルを示す図である。

【図8】第1の負荷分布を示す図である。

【図9】第2の負荷分布を示す図である。

【図10】第3の負荷分布を示す図である。

【図11】第4の負荷分布を示す図である。

【図12】第4のスケジューリングの契機を示すフローチャートである。

【図13】スケジューリング処理のフローチャートである。

【図14】再配置リスト生成処理のフローチャートである。

【図15】最適化処理のフローチャートである。

【図16】割当処理のフローチャートである。

【図17】情報処理装置の構成図である。

【図18】記録媒体を示す図である。

【図19】第1のシミュレーション結果を示す図である。

【図20】第2のシミュレーション結果を示す図である。

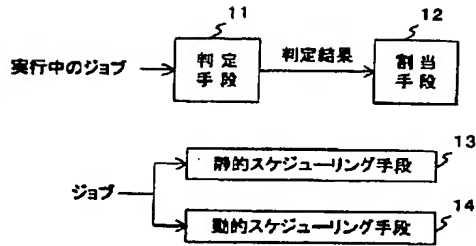
【図21】第3のシミュレーション結果を示す図である。

【符号の説明】

- 11 判定手段
- 12 割当手段
- 13 静的スケジューリング手段
- 14 動的スケジューリング手段
- 21 コンピュータ
- 22 ネットワーク
- 23 外部補助記憶装置
- 31 スケジューラ
- 32 キューイングシステム
- 33 待ち行列
- 41 並列ジョブ
- 42 ジョブ
- 51、91 CPU
- 52、92 メモリ
- 61 ジョブ情報テーブル
- 71 ジョブのメモリサイズ
- 72 ジョブの実行経過時間
- 73 ジョブの実行コスト履歴
- 74 データ入出力性能
- 75 データ転送性能
- 81 再配置リスト
- 82 移動しないジョブ
- 83 通常ジョブ
- 84 移動するジョブ
- 85 凍結対象ジョブ
- 86 ジョブの凍結処理
- 93 入力装置
- 94 出力装置
- 95 外部記憶装置
- 96 媒体駆動装置
- 97 ネットワーク接続装置
- 98 バス
- 99 可搬記録媒体
- 100 データベース

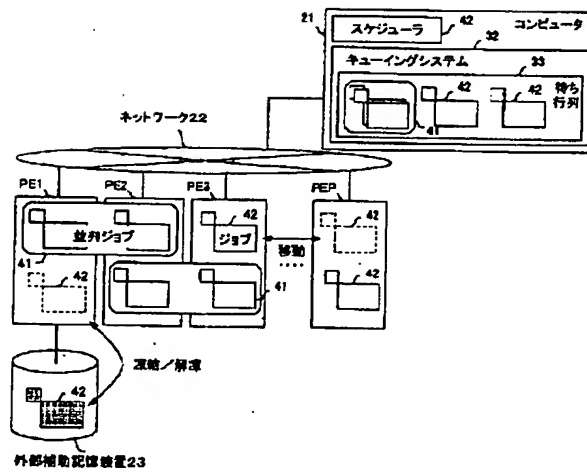
【図1】

本発明の原理図



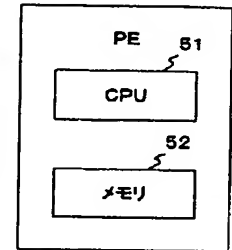
【図2】

並列計算機システムの構成図



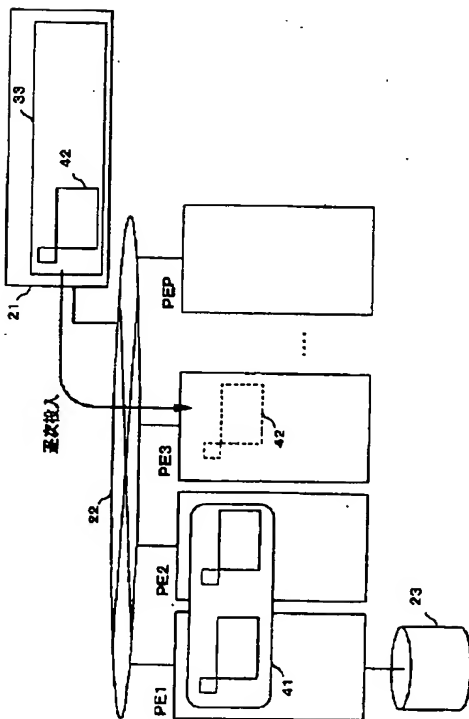
【図3】

PEの構成図



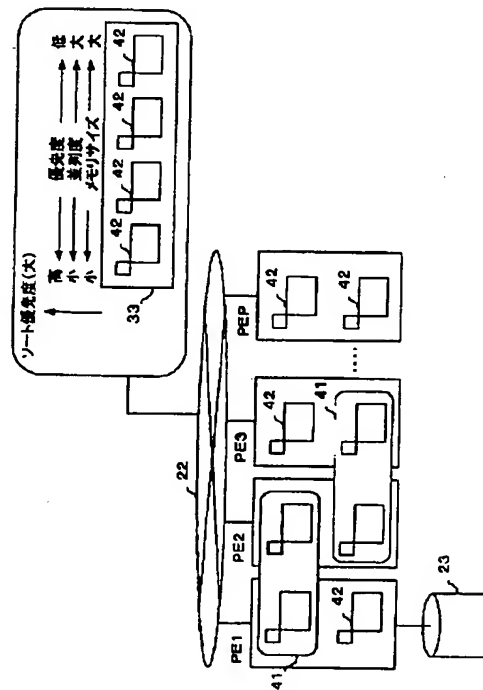
【図4】

第1のスケジューリングを示す図



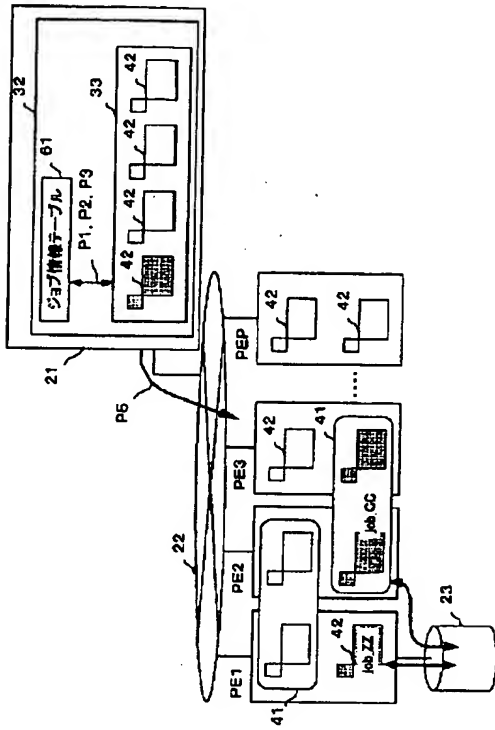
【図5】

第2のスケジューリングを示す図



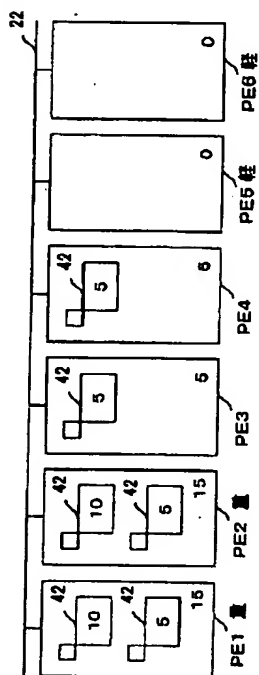
【図6】

第3のスケジューリングを示す図



【図9】

第2の負荷分布を示す図



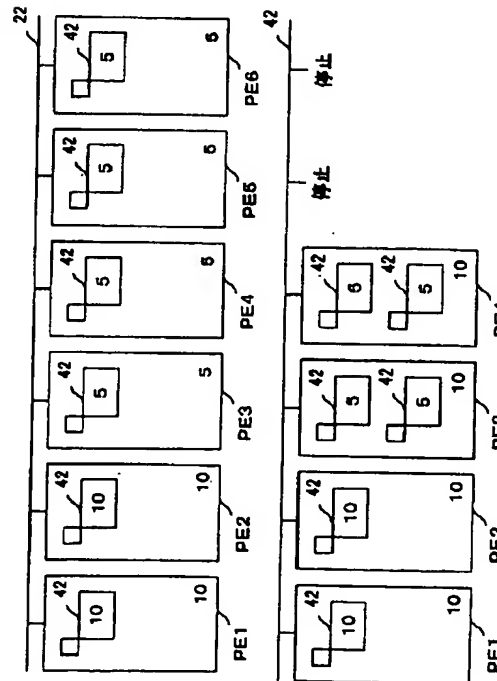
【図7】

ジョブ情報テーブルを示す図

ジョブ名	ジョブID	優先度	並列度	メモリサイズ	ジョブ投入時刻	アイドル時間
job.XX	105	1	6	100	990802083000	0
job.AA	110	1	8	100	990802080000	0
job.BB	107	1	6	60	990802084500	0
job.YY	98	2	8	100	990802080000	0
...						
job.ZZ	100	10	1	50	990802081000	5
job.CC	123	10	2	75	990802081500	10

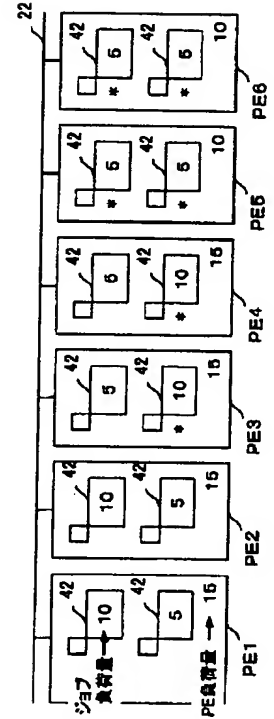
【図10】

第3の負荷分布を示す図 第4の負荷分布を示す図



【図8】

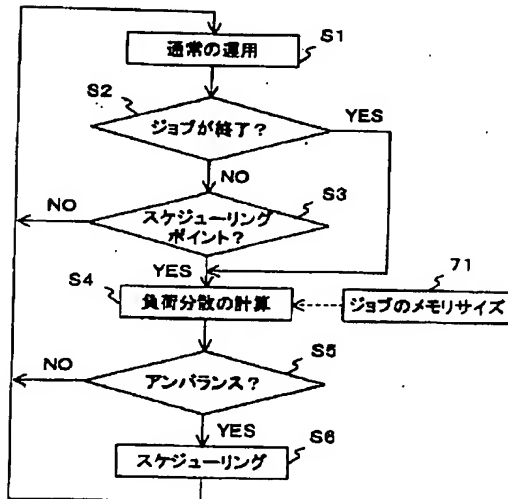
第1の負荷分布を示す図



【図11】

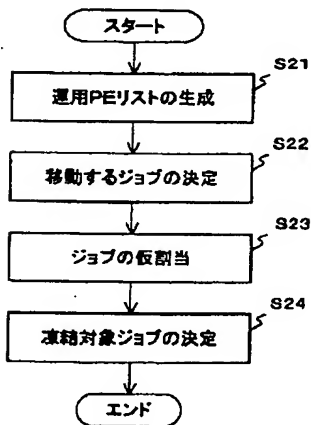
【図12】

第4のスケジューリングの契機を示すフローチャート



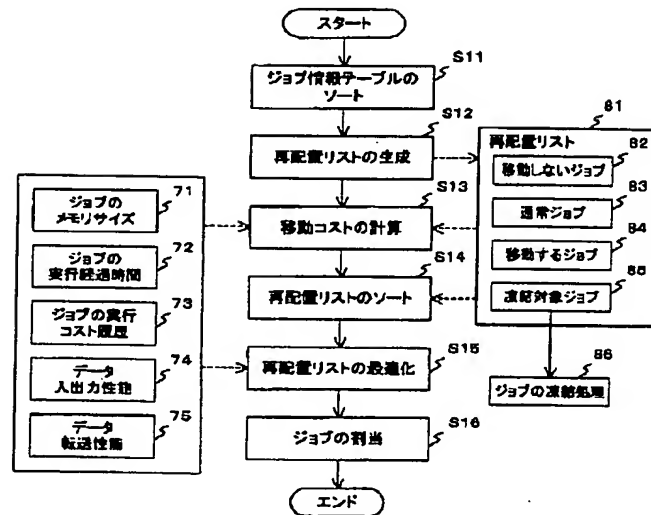
【図14】

再配置リスト生成処理のフローチャート



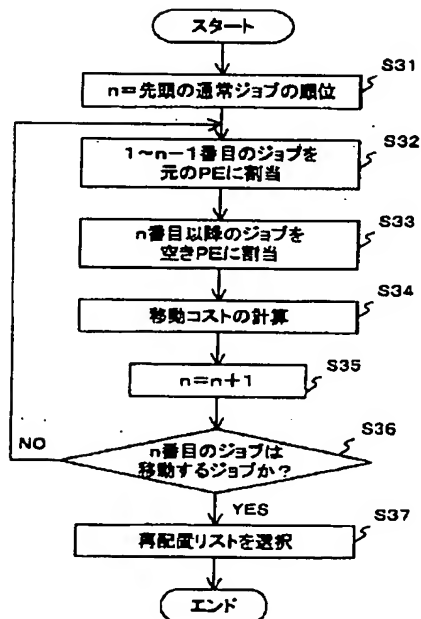
【図13】

スケジューリング処理のフローチャート



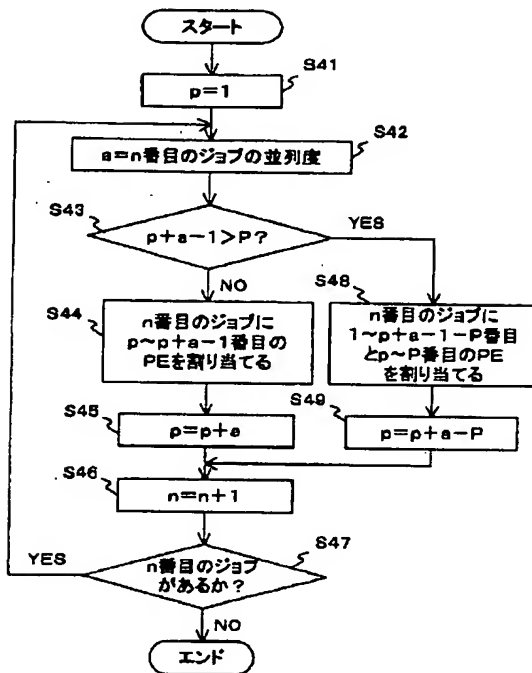
【図15】

最適化処理のフローチャート



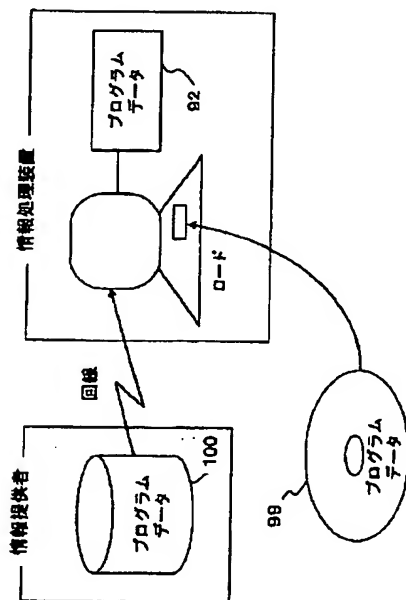
【図16】

割当処理のフローチャート



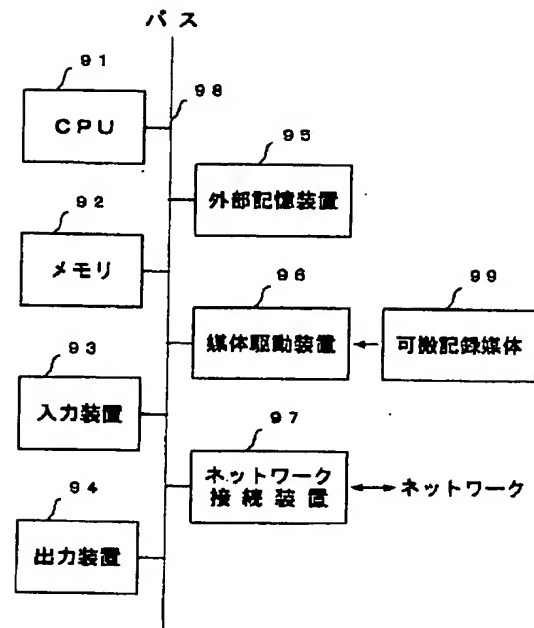
【図18】

記録媒体を示す図



【図17】

情報処理装置の構成図



【図20】

第2のシミュレーション結果を示す図

	実行回数	待ち時間 スケジューリング	実行時間スケジューリング			
			δO=25	δO=50	δO=75	δO=100
総実行時間 [sec]	6.12 × 10 ⁶	5.62 × 10 ⁶	4.89 × 10 ⁶	4.10 × 10 ⁶	5.01 × 10 ⁶	5.01 × 10 ⁶
性能比 (逐次=1)	1	0.91	0.80	0.67	0.82	0.82
メモリ使用率 [%]	25	36	55	57	52	52

【図19】

第1のシミュレーション結果を示す図

	逐次実行	待ち行列 スケジューリング	階層的スケジューリング		
			$\delta 0=25$	$\delta 0=60$	$\delta 0=75$
総実行時間 [sec]	8.56×10^6	6.22×10^6	5.48×10^6	5.62×10^6	6.71×10^6
性能比(逐次=1)	1	0.72	0.64	0.66	0.68
メモリ使用率 [%]	23	34	52	49	48

【図21】

第3のシミュレーション結果を示す図

	逐次実行	待ち行列 スケジューリング	階層的スケジューリング		
			$\delta 0=25$	$\delta 0=50$	$\delta 0=75$
総実行時間 [sec]	5.68×10^6	5.49×10^6	5.12×10^6	5.03×10^6	4.78×10^6
性能比(逐次=1)	1	0.97	0.90	0.88	0.84
メモリ使用率 [%]	21	33	48	45	50